CS 170, Spring/1998
Midterm #2
Professor J. W. Demmel

Problem #1 (12 points)

The following describes an execution of MAKESET, UNION, and FIND operations on a set of 10 elements, labeled 1 through 10. MAKESET assigns rank 0 to an element, and UNION breaks ties by putting the tree whose root has the larger label as the parent of the other.

```
for j = 1 to 10
MAKESET(j)
endfor
UNION(1,2); UNION(1,3); UNION(4,5); UNION(4,6); UNION(1,6); UNION(7,8); UNION(9,10);
UNION(7,10); UNION(3,8); FIND(4); FIND(3);
```

- Give the tree from executing the above steps using union-by- rank *with no* path-compression. Be sure to label the nodes in the *final* tree, including their *final* ranks.
- Give the tree from executing the above steps using union-by- rank *with* path-compression. Be sure to label the nodes in the *final* tree, including their *final* ranks.

We recommend that you draw the intermediate trees for partial credit.

Problem #2 (16 points)

In this question we will consider how much Huffman coding can compress a file F of m characters taken from an alphabet of $n=2^s$ characters x_0, x_1, x_2,...,x_{n-1}.

- How many bits does it take to store F without Huffman coding?
- Suppose m = 1000 and n = 8, with characters 0,1,2,3,4,5,6, and 7. Give an example of a file F (a string of 1000 digits from 0 through 7) in which every character x_i appears at least once, which compresses *the most* under Huffman coding. How many bits does it take to store the compressed file?
- Let $f(x_i)$ denote the frequency of x_i, i.e. the number of times x_i appears in F. Prove that there exist frequencies $f(x_i) > 0$ such that the number of bits needed to store F without Huffman coding is (lower bound) log n times the number of bits to store F when it is Huffman encoded. You can assume that the length of the file m, is much larger than n. Be sure to exhibit the bit patterns representing each character, both with and without Huffman coding, as well as explicit formulas for each $f(x_i)$.

Problem #3 (20 points)

In class we derived the FFT for vectors of length n a power of two. In this question we will derive the FFT for $n = 3^s$, a power of three.

- Let p(z) = (summation from j=0 to n-1 of) $p_j z^j$ be a polynomial of degree at most n - 1, where $n = 3^s$. Show that p(z) can be written as the sum

  $p(z) = p0(z^3) + zp1(z^3) + z^2 p2(z^3)$ (1)

  where p0(z'), p1(z'), p2(z') are each polynomials of degree at most (n/3) -1. Be sure to explicitly exhibit the coefficients of each polynomial.

- Let $w = e^{2(pi)i/n}$, $i = (-1)^{1/2}$, be a primitive n-th root of unity. Using equation (1), show that you can evaluate p(z) at the n points $w^0, w^1, {}^2, ..., w^{n-1}$, *given* the values of the 3 polynomials p0(z'), p1(z'), and p2(z') at the n/3 points $w^0, w^3, w^6, w^9 ..., w^{n-3}$. You should write down a loop that evaluates $p'_j = p(w^j)$, for j = 0 to n -1, in terms of the values of p0(z'), p1(z'), and p2(z').

- Write a recursive subroutine for evaluating p(z) at w^j, j = 0,..., n-1. Use your answer from the previous part in your answer.

- What is the complexity of your recursive subroutine? You should write down a recurrence for the complexity T(n), justify it, and quote a theorem from class to solve it.

Problem #4 (18 points)

Give a divide and conquer algorithm for the following problem: you are given two sorted lists of size m and n and are allowed unit time to access the j-th element of each list. Give an *O*(log m + log n) time algorithm for computing the k-th largest element in the union of the two lists.

Give a recurrence relation for this problem and determine its complexity. Make sure you justify your recurrence relation and show your work when solving it. Hint: binary search.

Problem #5 (9 points)

True or false? No explanation required, except for partial credit. Each correct answer is worth 1 point, but 1 point will be *subtracted* for each wrong answer, so answer only if you are reasonably certain.

a. In a UNION-FIND data structure, a root node of rank three can have exactly one child.

b. In UNION-FIND, the rank of a node can be equal to the rank of its parent.

c. In UNION-FIND, FIND with path compression can take a maximum of log(n) steps, where n is the number of elements.

d. The algorithm for computing a Huffman code is an example of a greedy algorithm.

e. The solution of $T(n) = 9T(n/2) + n^3$ is Theta(n^8).

f. The solution of $T(n) = T(n-1) + n^4$ is $O(n^6)$.

g. The solution of $T(n) = T(n - 1000) + n^2$ is $O(n^3)$.

h. The product $w^1 w^2 w^3 \ldots w^n$ of the n- th root of unity is either 1 or -1 for all n.

i. The coefficients of the polynomial p(x) = (Summation from j=0 to n-1 of) $p_j x^j$ of degree at most n - 1 are uniquely determined by the values $p(x_j)$ of the polynomial at n arbitrary points $x_0, \ldots, x_{n-1}$.

Problem #5 (9 points)